AUTONOMOUS AND MOBILE AGENTS IN DISTRIBUTED NETWORK MANAGEMENT AND MONITORING SYSTEM

UMA SHANKER,
Innovative Network Applications,
IBM Scientific Center,
18 Vangerowstrasse, Heidelberg 69115, Germany
Email: uma@de.ibm.com,
Tel: ++49-6221-594560, Fax: ++49-6221-593300

Abstract
Distributed object technology provides optimum architecture for Internet and intranet web applications. This paper presents a generic architecture solution for the Network Management System. Central to the distributed network management is the broker which provides access to the real-time information from the network nodes. It explains how the broker is used to manage and access management data on real-time basis. Finally we talk about the autonomous and mobile agents in the given architecture. It explains use of design patterns like broker, proxy etc. and some details and experiences from the ongoing project are described. Areas for future research are also explored.

Keywords: Agents, Network Management, CORBA, Java, SNMP

1 INTRODUCTION

In today's growing internet it's very important to have up-to-date information about your systems components and it becomes more important if you are internet provider. Increasing size and complexity of network and the need to support all sorts of advanced services in a reliable and cost-effective manner pose a major challenge to network management.

The traditional approach to develop large complicated network management/monitoring systems and applications was to build them using a uniform platform-specific architecture, based on openly defined management interface such as defined by SNMP[6] management framework. In the traditional model, a network management application's user interface is provided by an
appropriate data-driven protocol to a windowing or other display (like in Sun-Net Manager[2] from Sun). Recent advances in technology such as WWW[3] browsers and services, corporate intranet, and executable contents such as provided by Java[7], introduce the possibility for a more network-centric approach to the development of management applications, that allows organisations to leverage their existing Network Management System (NMS) solutions with generic, highly scalable and customizable Web-based front end. Network management is data-based. Vast amount of information (especially in large, complex networks) are collected by the network agents[1] and sent to the manager site. Manager site collects network performance, status and configuration information, maintains historical and statistical data, handles events and reports. All this information, which explodes in size with network complexity and size augmentation, need not only be stored efficiently but it must be enriched with powerful data management features that allow the realization of demanding, high level management functions like temporal reasoning, decision-making, planning etc. Additional functionality is also required in large multiple domain network environments.

As we come into Web-based front end system, there are several places which are very critical to the performance and reliability of the system. Very first point is, how to handle data in the reliable and easy way? Security is yet another important issue. We propose the Distributed Network Management and Monitoring System (DNMMS) architecture, which is based on Java[7] and Object Request Broker[11]. We are using Voyager from object space, as in addition to the fundamental ORB features it also support mobile objects, autonomous agents, events and listeners, database independent persistence, directory services etc. On the other side, we are using Java due to its ability to load classes into a virtual machine at run time. This capability enables infrastructures to use mobile objects and autonomous agents as another tool for building distributed systems. DNMMS architecture is shown in figure 1.

2 BACKGROUND

Various Tools/Systems are available today in the market for System Management and Monitoring. Please note that we are using the term System management and not Network Management, as in todays distributed environment, Internet providers are not only interested in there network components like routers, but also in there web-applications like http and proxy servers and they like to have common interface for all the management applications. Generally these tools runs on different Platforms and generally not
compatible with each other. Extension/upgrading is also very difficult. Now a days one of the common way to provide the statistics about the systems like web, routers, nodes, etc. using web-front end is to collect the statistics information at regular time interval at central location, where the data is formatted(mainly by CGI) for the administrator to display the data in the web browser. But CGI is a state-less technology, so we don’t have any persistent state during the browsing. In general all of the data, some of them are never used by any body, is transmitted from the nodes to the central location. There is no connection with the system once data is retrieved. If we want to have further special query about some node/machine, we have to use some other system or all the queries are done at the central location but not at the node/machine, which makes that tool not real-time and hence not 100% reliable which is necessary for the network components(routers/proxy/web servers) of the Internet Service Providers. Next section explains architecture of the Java/Corba based DNMMMS system and also the extension and integration of existing tools.

3 OVERVIEW OF DNMMMS

3.1 DNMMMS Architecture

Central to the DNMMMS architecture are two components CountryBase and Embassy. CountryBase is implemented according to the Broker Pattern[10].

109
On one side it collects management information from the network nodes and provides facilities for the real-time management data, on the other side it is used by the front end to present management data in the visual (graphical/text) format. Please note that front end can be application or applet, which can access CountryBase by DNMS API.

Embassy is based on the Proxy Pattern. It is situated at network nodes which provides data from that node according to the standard interface. Interface to the Embassy is described in the IDL format, which allows its implementation in different programming languages like Java, C++. As Embassy will be running on different platforms, it allows to have optimal implementation for that platform. It gives possibilities to integrate existing management systems. Each network node can have several management objects. Every management object is described by its management object identifier (MOI). CountryBase requests management data from Embassy by providing MOI. MOI completely specifies manageable resource inside the DNMS.

You can compare the DNMS as a country, who has there embassies located at other foreign countries (different servers, routers). Most important is that Embassies are not engaged with the foreign countries government. They just provide the required information within the protocol set by the two countries. Inside DNMS every object like embassy, agent, etc. have unique identification numbers called GUID (16-byte globally unique identifier).

### 3.2 CountryBase overview

CountryBase is based on the Broker Pattern [8]. It is responsible for coordinating communication, such as forwarding requests, database and directory services as well as for transmitting results and exceptions. Instead of focusing on low-level inter-process communication, it allows to access distributed services simply by sending message calls to the appropriate Embassy. In addition, the CountryBase architecture is flexible, in that it allows dynamic change, addition, deletion, and relocation of MOI. CountryBase just transmits requests made by the client to the appropriate Embassy. In general CountryBase is responsible for:

- Registration of embassy
- Transfers of messages
- Error recovery
- Interoperation with other brokers
- Locating embassies
- Directory services
- Agent management
- Storage of management data
CountryBase has a central distributed database which stores all the management data collected from the different embassies and also data to manage CountryBase itself. Every information exchange between a client and Embassy passes through the broker.

3.3 Embassy Overview

Embassy is based on the Proxy pattern[9]. It is installed at every network node and it provides data from that node according to the standard interface. Embassy during the initialization process registers himself with the CountryBase. Embassy can be started through remote machines as all the classes needed for the Embassy can be automatically downloaded. This allows easy installation and management.

Client make request by providing MOI to the Embassy. After receiving request Embassy works according to the local implementation and provides back the result. General format of MOI is MOI://PROTOCOL/VERSION/OI/ where PROTOCOL is a name of protocol like SNMP, VERSION is a version of the MOI implementation and OI is a Object Identifier, eg. in case of SNMP OI can be 1.3.6.1.2.1.7.1 for the UdpInDatagrams. Please note that this format on the one side allows to use industry standard like SNMP but are not only limited to that. All informations related to MOI are available through the CountryBase Directory Service under [/CountryNet/MOI]. Example of MOI is: MOI://SNMP/1.0/1.3.6.1.2.1.7.1/.

Interface to the Embassy is described in the IDL[5] format, which allows its implementation in different programming languages like Java, C++. As Embassies may will be running on different platforms, so there implementation can be different. Working of DNMMS using ORB and MOI is shown in figure 2.

4 AGENTS IN DNMMS

4.1 Agents in CountryBase

DNMMS Agents are the voyage's mobile objects, they can move at run time from one virtual machine to another. In this way, agents can act independently on the behalf of a client, even if the client is disconnected or unavailable. CountryBase has central directory to hold every neccessary information to operate Agents in the DNMMS. Each Agent in the DNMMS is associated with unique identification number called AUID, which is
a GIUD of that Agents Virtual Reference[4]. All the informations are stored in central directory on CountryBase in [/Agent]. As the agent proceeds from embassy to embassy agent related data are stored in this central directory. This provide one point management of all the agents in DNMMS. This provides facilities to check status of different agents in DNMMS. Further enhancement can be done by providing agents related information on date, status, last logged, etc. basis. CountryBase is a central location to execute commands like start, stop, wait etc. on the agents.

4.2 DNMMS System Agents

There are severals DNMMS agents which are started at the CountryBase creation time. These agents are DNMMS System Agents and are generally live forever status[4]. They are used to manage DNMMS itself.

4.3 DNMMS Management Agents

DNMMS management agents are the agents which can be used to monitor status of the network nodes. These agents are completely autonomous. It means that they can control themselves in different situations. An agent is a special object type. An agent has autonomy. An autonomous object can be programmed to satisfy one or more goals, even if the object moves and lost contact with his creator. Using voyager DNMMS supports autonomous and mobile agents, mobility is the ability to move independently from one device to another on a network. When agent moves to new location , he leaves behind a forwarder to forward messages.
If the agent want to have high-speed conversation with remote object, the agent can move to the object and then send it local Java messages.

Autonomous agent in DNMMS is useful for many reasons, for example:

- If a task must be performed independently of the computer that launches the task, a mobile agent can be created to perform this task. Once created, the agent can move into the network and complete the task in a remote program.

- If the periodic monitoring of a remote object (in our case object return back the status of himself at the CountryBase) is required, creating an agent that meets the remote object and monitors it locally is more efficient than monitoring the device across the network.

Agent knows nothing about the remote location address and other useful informations which is needed to access that location. Agent gets all these information from the CountryBase Directory Service. Once agent has virtual reference of the remote embassy he can use it to find out other informations like, GUID, network address, port and so on. As a example, there can be agent which goes to the target Embassy and tries to monitor the tcp segments received and send. If it increases 2000 segments/sec then agent reports to the output. Please note that any other action can be taken on this event. Agent can finally move to the CountryBase or can be parked at the Embassy or can send some notification and so on.

5 CONCLUDING REMARKS

The growing complexity of the network infrastructure requires a reliable and real-time management system. A web-based, network centric design gives the flexibility to offer such a solution, and additionally allows one to scale up the solution because of its genericity. We believe that architecture presented is one of the first such application of its kind. But there are still some important open issues like security, performance and workload.

6 ACKNOWLEDGMENTS

Thanks should go to Dr. B. Lamparter for interesting comments and to Dr. J. Rueckert, Prof. P. Schmidt and Prof. W-D. Oberhoff for their encouragement.
References


Author

Uma Shanker works as a Guest Scientist at IBM Scientific Center, Heidelberg, Germany. Currently he is working on Caching and Proxy servers. His interest includes distributed computing, CORBA, web caching, Java and network management.